УДК 004.8:004.93

Н.А. Костромов, И.В. Бекетова, С.Л. Каратеев
ФГУП «Государственный научно-исследовательский институт
авиационных систем», г. Москва, Россия
Россия, 125319, г. Москва, ул. Викторенко, 7

Формирование пространства признаков и их извлечение из изображений для систем индексации документов

N.A. Kostromov, I.V.Beketova, S.L. Karateev
The Federal State Unitary Enterprise "State Research Institute of Aviation Systems", c. Moscow, Russia
Russia, 125319, Moscow, Viktorenko street, 7.

Formation of Feature Spaces and their Extraction from the Images for Documents Indexing

Н.А. Костромов, І.В. Бекетова, С.Л. Каратєєв
ФГУП «Державний науково-дослідний інститут авіаційних систем»,
Росія, м. Москва
Росія, 125319, м. Москва, вул. Вікторенко, 7

Формування простору ознак і їх добування із зображень для систем індексації документів

Предложены оригинальные алгоритмы анализа изображения для формирования вектора признаков. Вектор признаков сформирован из геометрических признаков и составленных на основе системы правил комбинированных признаков. Вектор признаков используется для поиска областей со специфической графической информацией на изображениях документов. Моделирование на тестовой выборке изображений документов показало устойчивость и эффективность работы алгоритмов.
Ключевые слова: индексация, излечение признаков, машинное обучение

The original algorithms of image analysis for formation of feature vector are proposed. Feature vector is formed from the both type of features, geometric features and secondary complex features compiled on the basis of the logical rules. Feature vector is used to find areas with specific information on the graphic images of documents. Simulation on a test set of images of documents has shown the stability and efficiency of the algorithms.
Key Words: indexing, feature extraction, machine learning.

Запропоновані оригінальні алгоритми аналізу зображень для формування вектора ознак. Вектор ознак сформований з геометричних ознак і складених на основі системи правил комбінованих ознак. Вектор ознак використовується для пошуку областей зі специфічною графічною інформацією на зображеннях документів. Моделювання на текстовій виборці зображень документів показало стійкість й ефективність роботи алгоритмів.
Ключові слова: індексація, добування ознак, машинне навчання

Введение

При использовании систем автоматического анализа документов возникают задачи обнаружения документов, содержащих специфическую графическую информа-

цию. Существует ряд документов, в которых содержание подкрепляется наглядными образами при помощи графических моделей в форме структурных схем или блок-схем. Сложность задачи обнаружения подобных документов состоит в том, что в отличие от изображений таблиц или других стандартных форм изображения графических моделей рассматриваемого типа, как правило, не имеют стандартной структуры, специфических ключевых слов, символов и графических объектов, по которым можно определить, что данное графическое изображение является структурной схемой или блок-схемой.

Структурные схемы и блок-схемы отличаются широким разнообразием состава и параметров отображающих их геометрических объектов – геометрических фигур, линий и графических символов. Примеры изображений документов, содержащих блок-схемы и структурные схемы, приведены на рис. 1.

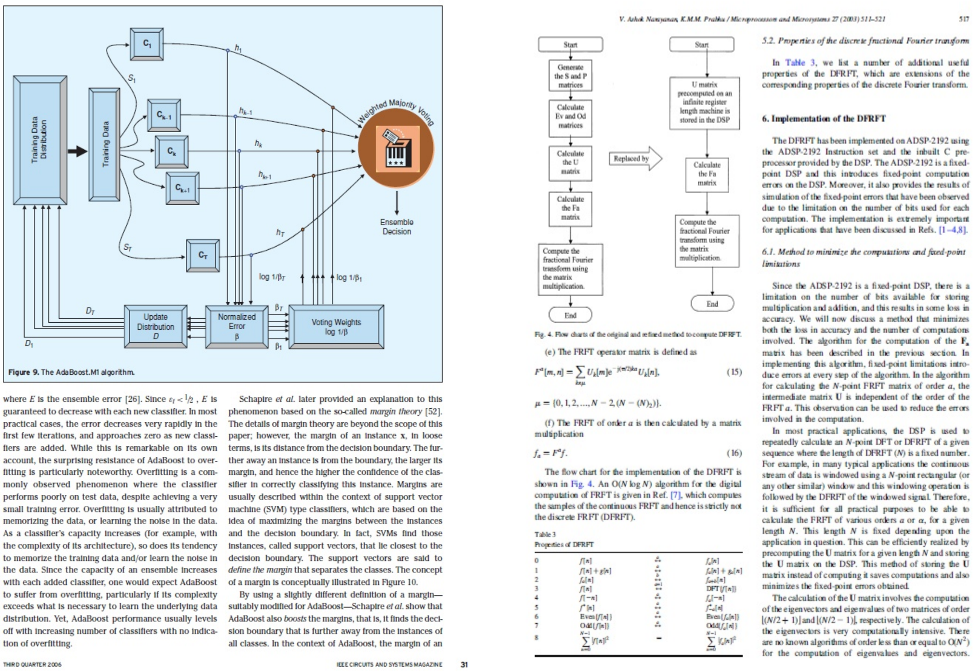


Рисунок 1 – Примеры изображений документов, содержащих блок-схемы и структурные схемы

Важно определить наиболее существенные признаки рассматриваемой графической модели. Данная графическая модель строится при помощи ограниченного набора графических элементов (линии, прямоугольники, ромбы, овалы и т.д.), часто дополняемых короткими текстовыми строками или отдельными графическими символами. Часто изображения содержат шумы и искажения, вызванные процессами получения изображений (сканирование, видеосъёмка) и сжатия полученных изображений.

Достаточно полный обзор современных методов анализа структуры документов приведён в [1], [2]. Методы, используемые для обнаружения изображений графических объектов, можно разделить на три основные категории:

– обнаружение на основе априорной информации о структуре;

– эвристические методы;

– статистические или оптимизационные методы.

Для методов обнаружения на основе априорной информации, как правило, характерны удовлетворительные показатели обнаружения, однако обобщающие способности методов данного типа ограничены.

Эвристические алгоритмы используют заранее определённые наборы логических или предварительно определенных синтаксических правил для получения решения об обнаружении. Наборы эвристических правил обычно строятся на базе системы локальных признаков. Использование эвристик нередко требует сложной пост-обработки для окончательного принятия решений.

Статистические или оптимизационные алгоритмы, как правило, обходятся без априорной информации или используют свободные параметры, значения которых формируются в процессе обучения [3], [4].

Структура алгоритма

Общая структура алгоритмов представлена на рис. 2. Таким образом, выделение признаков можно разделить на следующие этапы:

1) выделение текста в документе;
2) создание бинарного препарата документа;
3) выделение признаков;
4) генерация пространства признаков.

Остановимся подробнее на каждом из этапов.

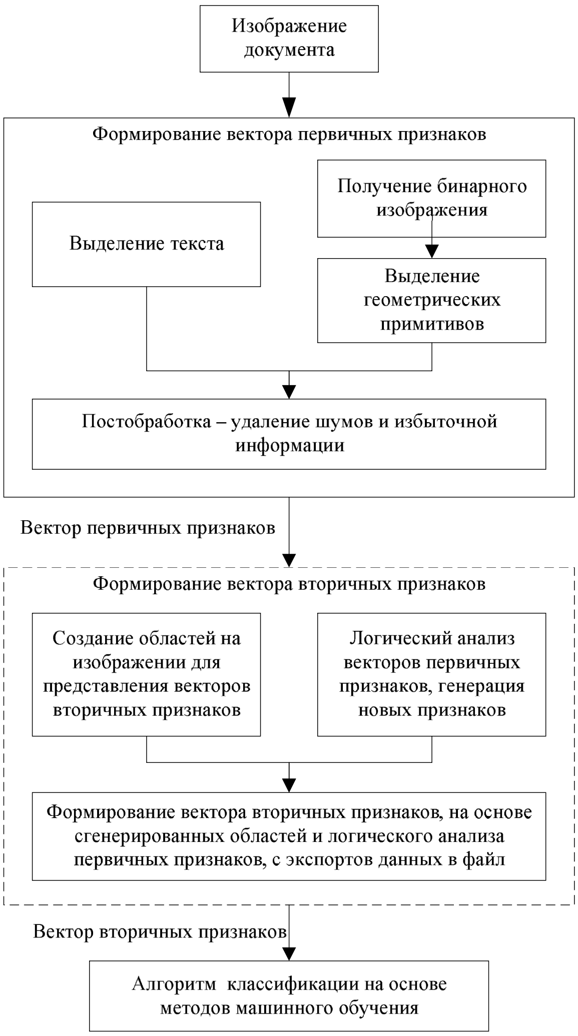


Рисунок 2 – Общая схема формирования вектора вторичных признаков для индексации документа

На первом этапе происходит выделение областей (маски) текста на изображении. Выделение текстовых строк и формирование маски текста на изображении выполняется путем классификации скользящим окном на основе разработанного авторами оригинального алгоритма адаптивного бустинга.

На втором этапе происходит бинаризация изображения (рис. 3). Перед бинаризацией изображение преобразуется в оттенки серого. Затем по всему изображению проходит окно (ядро низкочастотного фильтра), после чего из усредненных соответствующих пикселей вычитаются значения соответствующих пикселей исходного изображения (1). Если их разница отрицательна, то значение пикселя считается равным 0, иначе пиксель принимает значение, получившееся при вычитании (2). Затем выбирается порог бинаризации. В качестве порога бинаризации может использоваться комбинированный порог:

$$I'(x,y)=\sum_{i}^{M}\sum_{j}^{M}G(i,j)I(x+i,y+j)-I(x,y) \quad (1)$$

$$I''(x,y)=\begin{cases} I'(x,y),\ если\ I'(x,y)>0 \\ 0,\ если\ I'(x,y)\le 0 \end{cases} \quad (2)$$

$$I_{bin}(x,y)=\begin{cases} 1,\ если\ I''(x,y)\ge I_t \\ 0,\ если\ I''(x,y)<I_t \end{cases} \quad (3)$$

где I – изображение в оттенках серого, $I(x,y)$ – пиксель изображения, M – размер ядра свертки, G – ядро свертки фильтра, $I_{bin}(x,y)$ – бинаризованный пиксель, I_t – порог бинаризации.

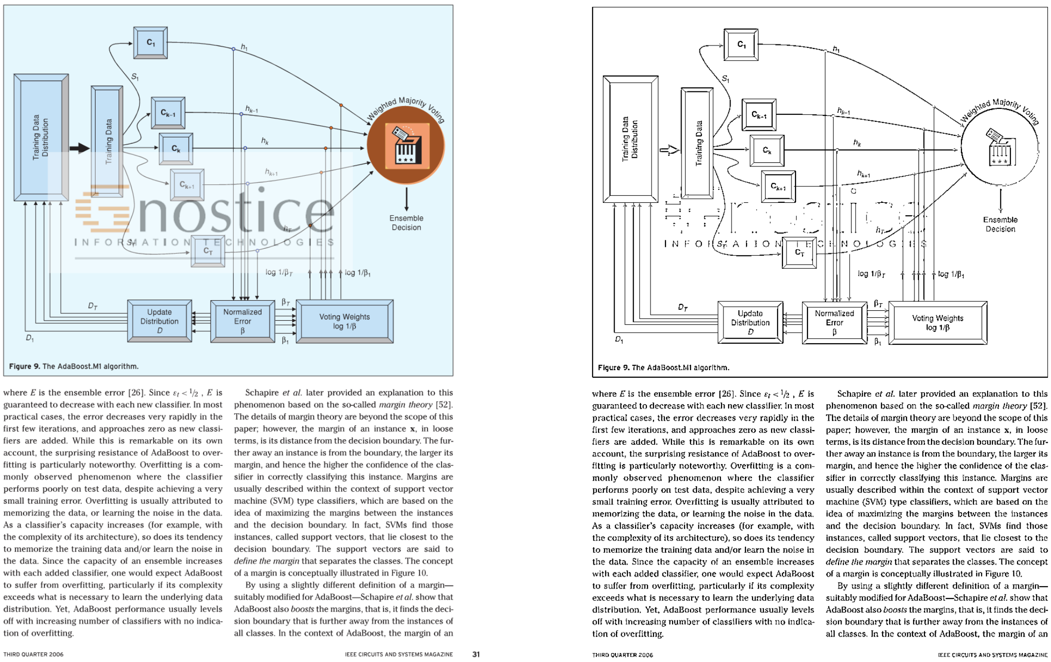


Рисунок 3 – Изображение до и после бинаризации

В результате должно получиться черно-белое изображение, как показано на рис. 3. Постобработка бинарного препарата включает в себя удаление отдельных точек и операции морфологического утоньшения. Далее возможно удаление областей изображения с выделенным на первом этапе текстом.

На третьем этапе происходит выделение признаков, сначала простых (рис. 4), а затем на их основе объединение в комбинированные признаки – прямоугольники и связи между ними (рис. 5). Простые геометрические признаки представляют собой горизонтальные и вертикальные линии. Эти линии обладают следующими атрибутами: длина, стиль (гладкая, штрихпунктирная), связность с другими линиями (пересечение или соединение). Линии обнаруживаются путем перебора пикселей по горизонтали

для горизонтальных линий и пикселей по вертикали для вертикальных линий. Алгоритм анализирует линию на стиль начертания – гладкий или штрихпунктирный. Псевдокод представлен в листинге 1.

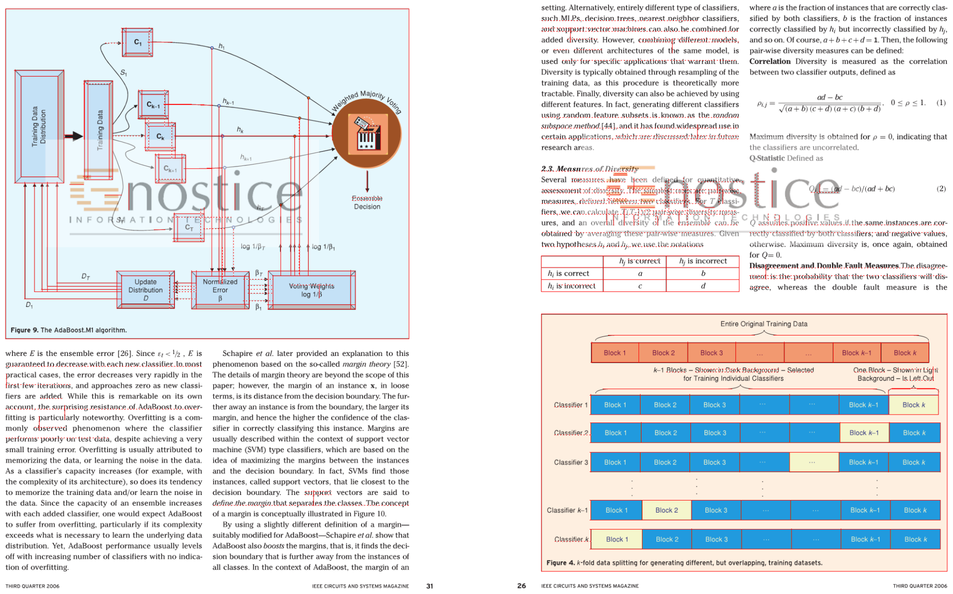


Рисунок 4 – Изображения с обнаруженными линиями

```
procedure GetLineFromImage(
                imgBin2Int[,]: Integer; //Массив пикселей бинаризированного изображения
                W_img,H_img: Integer;     //Ширина и высота изображения в пикселях
                LineList: List; //Список горизонтальных линий
                hmin: Integer; // Минимальная длина линии
                hspan: Integer; // Промежуток между линиями
                hmax: Integer; // Максимальная длина линии
                LineColor: Integer; // Цвет линии
                LDensity: Double; // Плотность линии
                DashLength: Integer); // Длина штриха
{
//Выделение горизонтальных линий
//Минимальная длина
lmin := hmin;
//Максимальная длина
lmax := hmax;
//Максимальные промежутки между соседними пикселями линии
lspan:= hspan;
//Цикл обработки по строкам для горизонтальных линий
For i := 0 To H_img - 1 Do
{
        //Сбросить счетчики
        clen := 0; cden := 0; cseq := 0; cemp := 0; cadd := 0;
        ptrLine^.StyleDraw    := ldsSolid;
        ptrLine^.StyleCross := lscNon;
        ptrLine^.StyleOrientation := lsoInclined;
        ptrLine^.UseForRect := false;
        ptrLine^.CountCross := 0;
        ptrLine^.LineLength := 0;
        cspan := 0;
        //Инициализируем прямую как "несуществующую"
        ptrLine^.Line.Left := -1;
        ptrLine^.Line.Right := -1;
        ptrLine^.Line.Top := i;
        ptrLine^.Line.Bottom := i;
```

```
        ptrLine^.StyleOrientation := lsoHorizontal;
        For j := 0 To Wimg - 1 Do
        {
                //Увеличиваем счетчик длины линии
                Inc(clen);
                //Если пиксель совпадает с цветом линии
                if(imgBin2Int[i,j] = LineColor)
                {
                        //Увеличиваем счетчик промежутков между кусками линии
                        if(cspan > 0)
                        {
                                Inc(cden);
                                if(cadd > DashLength)Then
                                        cseq := DashLength;
                        end;
                        Inc(cadd);
                        //Увеличиваем счетчик совпадающих пикселей линии
                        Inc(cseq);
                        // Если это первый совпавший пиксель,
                // то инициализируем начальную точку
                        If((ptrLine^.Line.Left = -1)And(cseq > DashLength))
                        {
                                clen := 0;
                                cemp := 0;
                                ptrLine^.Line.Left := j - DashLength;
                        }
                        //Т.к. совпал цвет линии, то присваиваем крайнее положение справа
                        if(cseq > DashLength)
                        {
                                ptrLine^.Line.Right := j + 1;
                                //change
                                cspan := 0;
                        }
                        //Т.к. цвет линии совпадает, то обнуляем счетчик "пустоты"
                        //cspan := 0;
                }
                //Цвет линии не совпадает
                else
                {
                        //Увеличиваем счетчик пустого пространства между отрезками линии
                        Inc(cspan);
                        Inc(cemp);
                        if(cspan > DashLength)
                                cadd := 0;
                }
                //Если промежуток между соседними точками больше разрешенного промежутка
                if(cspan >= lspan)
                {
                        // Проверяем не является ли линия пустой
                        if((ptrLine^.Line.Left <> -1)And(ptrLine^.Line.Right <> -1))
                        {
                                // Задаем ориентацию линии горизонтальной
                ptrLine^.StyleOrientation := lsoHorizontal;
                                // Проверяем линию на допустимость значений и
                                // проверяем тип начертания гладкая или штрихпунктирная
                                if(CheckLine(ptrLine, LDensity))
                                {
                                        // Добавляем линию
                                        LineList.Add(ptrLine);
```

```
                    // Создаем новую линию
                    New(ptrLine);
                }
                // Сбрасываем счетчики, т.к. превышено значение
                // пустоты между отрезками линии
                clen := 0; cden := 0; cseq := 0;      cemp := 0; cadd := 0;
                ptrLine^.StyleDraw   := ldsSolid;
                ptrLine^.StyleCross := lscNon;
                ptrLine^.StyleOrientation := lsoInclined;
                ptrLine^.UseForRect := false;
                ptrLine^.CountCross := 0;
                ptrLine^.LineLength := 0;
                //Инициализируем прямую как "несуществующую"
                ptrLine^.Line.Left := -1;
                ptrLine^.Line.Right := -1;
                ptrLine^.Line.Top := i;
                ptrLine^.Line.Bottom := i;
            }
        }
    }
    // Проверяем линию на допустимость значений и
    // проверяем тип начертания (гладкая или штрихпунктирная линия)
    if(CheckLine(ptrLine, LDensity))
    {
        // Добавляем линию
        LineList.Add(ptrLine);
        // Создаем новую линию
        New(ptrLine);
    }
}
```

Листинг 1 – Псевдокод нахождения горизонтальных линий в бинарном изображении

Далее эти линии вместе с маской текста образуют более сложные признаки. Выделяются прямоугольники из найденных линий, определяется включение в прямоугольниках других линий, других прямоугольников, текста, наличие и количество связей с другими прямоугольниками или линиями.

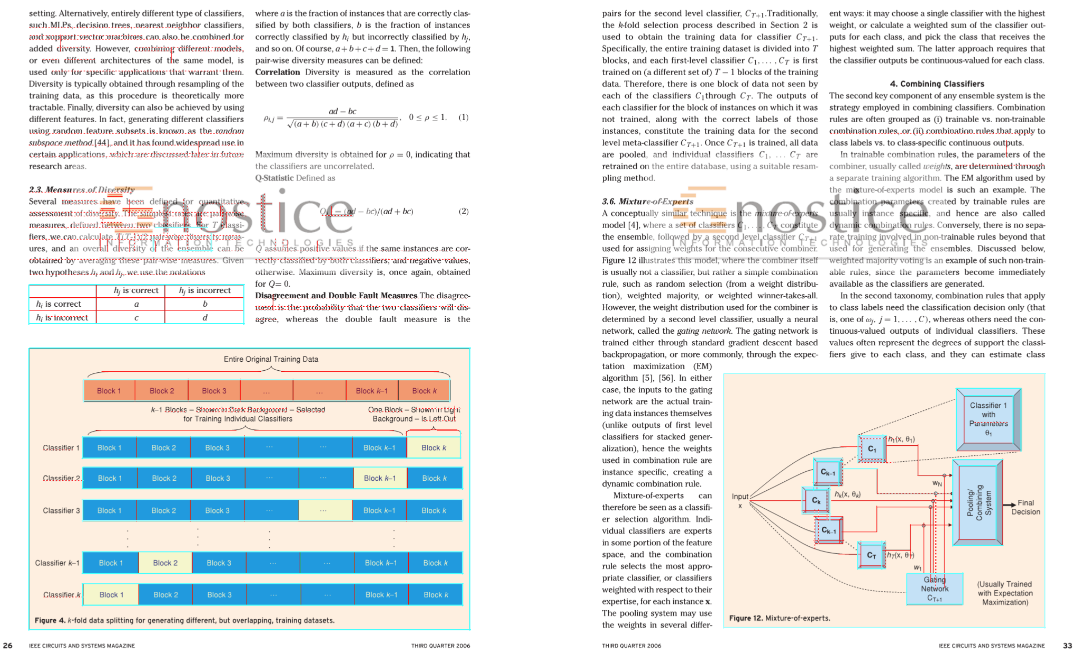


Рисунок 5 – Изображение с обнаруженными линиями и прямоугольниками

На четвертом этапе на основе полученных признаков происходит генерация системы признаков, на основе этих данных система обучается и принимает решения о наличии и месте расположения схем. Система признаков представляет собой общие данные и набор областей с единой структурой признаков. Например, изображение делится на сетку размером 4 на 4, тогда области, к примеру, имеют следующие размеры: 2х2, 2х3, 3х2, 3х3, 4х4 ячейки. Самой большой по размерам областью является все изображение. Пример псевдокода для генерации приведен в листинге 2.

```
void GenerateRect()
{
CellCount_X := 4;                // Количество ячеек по оси x
CellCount_Y := 4;                // Количество ячеек по оси y
Frame_min := 2;                  // Минимальный размер области в единицах сетки
Frame_max := Max(CellCount_X, CellCount_Y); //Максимальный размер области в единицах сетки
W_Cell := Ceil( W_img / CellCount_X);          // Вычисление ширины ячейки в пикселях
H_Cell := Ceil( H_img/ CellCount_Y);           // Вычисление ширины ячейки в пикселях
For ii := Frame_min To Frame_max Do
{
 For jj := Frame_min To Frame_max Do
 {
 For i := 0 To (CellCount_Y - ii) Do
 {
 For j := 0 To (CellCount_X - jj) Do
 {
 Rect := new Rect(0,0,0,0);  // Создание области с нулевыми координатами и размерами
 Rect.Left := i * W_Cell;                  // Вычисление левой границы области
 Rect.Right := Rect.Left + jj * W_Cell; // Вычисление правого угла области
 Rect.Top := j * H_Cell;                   // Вычисление верхней границы области
 Rect.Bottom := Rect.Top + ii * H_Cell;//Вычисление нижней границы области
 ListOfFrame.Add(Rect);     // Добавление области в список просматриваемых областей
 }
 }
 }
}
}
```

Листинг 2 – Псевдокод генерации областей
для последующего создания структуры признаков

Таким образом, все элементы признаков, которые будут включаться в сгенерированную область изображения, будут учитываться при вычислении признаков данной области.

Вектор признаков полного изображения включает в себя:

– количество прямоугольников;

– количество прямоугольников с текстом;

– количество линий в прямоугольниках;

– гистограмма длин горизонтальных и вертикальных линий (раздельно штрихпунктирные и гладкие), нормализованные под размер изображения.

Вектор признаков для анализируемых фрагментов изображения состоит из следующих элементов:

1. Координаты прямоугольной области.

2. Гистограмма длин горизонтальных и вертикальных линий (штрихпунктирные и гладкие), нормализованных на максимальный размер данной области.

3. Количество всех прямоугольников, включенных в эту прямоугольную область.

4. Количество прямоугольников, включенных в эту область, которые имеют текст.

5. Количество горизонтальных линий в прямоугольниках, включенных в данную область.

6. Количество полусвязных линий (линия имеет соединение с прямоугольником, только с одной стороны).

7. Количество полусвязанных прямоугольников (прямоугольник связан только с линией, но не с другими прямоугольниками).

8. Количество полносвязных линий (линии, которые своими концами соединены с прямоугольниками)

9. Количество полносвязных прямоугольников (прямоугольники, которые связаны друг с другом линией)

10. Количество прямоугольников внутри области и для каждого следующего признака:

а) количество всех линий этого прямоугольника,

б) количество всех горизонтальных линий,

в) наличие текста,

г) нормализованная площадь к площади области,

д) соотношение ширины к высоте.

Выводы

Для оценки эффективности предложенного метода было проведено моделирование работы оригинального классификатора на базе изображений документов. База изображений документов была разбита на две части: обучающую выборку и тестовую выборку. Обучающая база изображений образована из 600 изображений (300 – из положительных и 300 – из отрицательных примеров). Тестовая база изображений образована из 300 изображений (150 – из положительных и 150 – из отрицательных примеров).

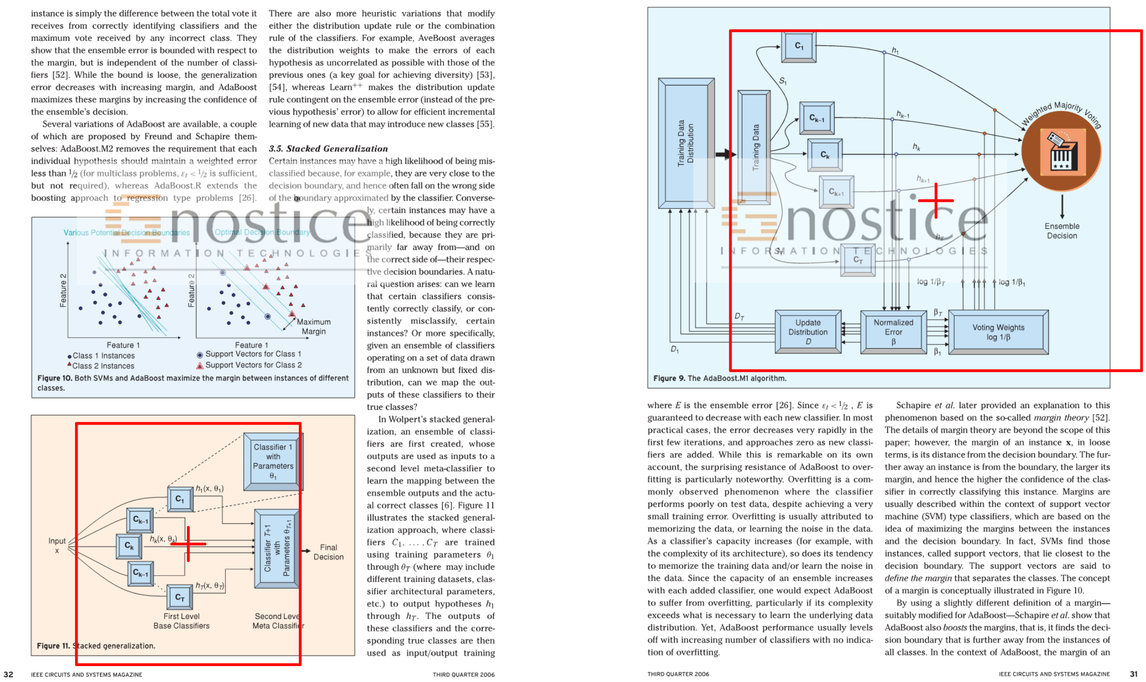


Рисунок 6 – Примеры работы алгоритма обнаружения и выделения

При моделировании на тестовой выборке изображений документов алгоритм обнаружения специфической графической информации показал устойчивость и эффективность работы. Получены следующие характеристики классификатора:

– вероятность правильной классификации (по тестовой выборке положительных примеров) составляет: 0.91;

– вероятность ложноположительной классификации (по тестовой выборке отрицательных примеров) составляет: 0.056.

Примеры работы алгоритма обнаружения специфической графической информации показаны на рис. 6, где рамками выделены области изображений документов, классифицированные как содержащие блок-схемы.

В качестве дальнейшего направления развития предполагается адаптация алгоритма к условиям работы с зашумлёнными изображениями, содержащими искажения. Для этого необходимо произвести соответствующие доработки алгоритмов бинаризации, выделения первичных признаков и формирования сложных признаков.

Литература

1. S. Mao. Document structure analysis algorithms: a literature survey / S. Mao, A. Rosenfeld, T. Kanungo // Document Recognition and Retreval X. – 2003. – Vol. 5011. – P. 197-207.
2. Shazia Akram. Document Image Processing. A Review / Shazia Akram, Mehraj-Ud-Din Dar, Aasia Quyoum // International Journal of Computer Applications. – 2010. –Vol. 10, № 5. – P. 35-40.
3. G. Nagy. A prototype document image analysis system for technical journals / G. Nagy, S. Seth, M. Viswanathan // Computer. – 1992. – Vol. 25. – P. 10-22.
4. A. Lemaitre. Contribution of Multiresolution Description for Archive Document Structure Recognition / A. Lemaitre, J. Camillerapp, B. Couasnon // Ninth International Conference on Document Analysis and Recognition. – 2007. – Vol. 01. – P. 247-251.

Literatura

1. S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis algorithms: a literature survey // Document Recognition and Retreval X. – 2003. – Vol. 5011. – P. 197-207.
2. Shazia Akram, Mehraj-Ud-Din Dar, Aasia Quyoum. Document Image Processing - A Review // International Journal of Computer Applications. – 2010. –Vol. 10, No. 5. – P. 35-40.
3. G. Nagy, S. Seth, M. Viswanathan. A prototype document image analysis system for technical journals // Computer. – 1992. – Vol. 25. – P. 10–22.
4. Lemaitre, J. Camillerapp, B. Couasnon. Contribution of Multiresolution Description for Archive Document Structure Recognition // Ninth International Conference on Document Analysis and Recognition. 2007. Vol. 01. P. 247-251.

RESUME

N.A. Kostromov, I.V.Beketova, S.L. Karateev

Formation of Feature Spaces and their Extraction from the Images for Documents Indexing

The article describes the formation of the original algorithms for generation of feature vectors for the detection of areas with specific the graphic images in the document image.

The process of formation of feature vector consists of two stages. In the first phase, we get a binary document image. Lines and text strings are allocated from the resulting image. The result is a set of primary attributes. In the second stage based on the analysis of a set of primary features, the combined features, i.e. rectangular areas, signs of the text, the signs of connectivity and nesting stand out. Feature vector is formed by combining the primary features and combined ones.

Trained with the given arrays of images, the system can detect the image of the document with the desired graphical information. The following characteristics of the classifier were obtained: the probability of correct classification (as a test set of positive examples) is 0.91; the probability of false positive classifications (on a test set of negative examples) is 0.056.

Статья поступила в редакцию 05.06.2012.